macromedia
FLASH
MX

# Програмиране с ActionScript MX

*/факултативен курс/*

*преподавател: Мая Стойкова*

# ЛЕКЦИЯ № 2

# Значение и използване на обекти в ActionScript MX. Работа с функции

# СЪДЪРЖАНИЕ

I. **Значение и използване на обекти в ActionScript MX.**

- Същност, свойства и употреба на обектите в ActionScript MX.
- Типове обекти.
- Използване на обекта Color.
- Използване на обекта Key за добавяне на интерактивност.
- Работа с обектите String и Selection

# СЪДЪРЖАНИЕ

**II. Работа с функции**

- Създаване на функции и добавяне на параметри към тях.
- Използване на локални променливи и създаване на функции връщащи резултат.

# Същност, свойства и употреба на обектите в ActionScript MX

**Обектите във Flash са елементи, които наподобяват обектите от реалния свят. Те са обособени в отделни типове. Можем да ги използваме даже, да изпълним задачи със самите тях. Например филмче, което записва или стартира друго movie.**

**Самият език ни осигурява множество предварително създадени обекти за различни нужди. Но в случай, че не може да ни послужат за някоя специална задача, имаме възможност сами да си създадем наш (потребителски) обект.**

**Обектите се дефинират с помощта на две основни характеристики:**

- свойства (полета)
- методи (функции)

# Същност, свойства и употреба на обектите в ActionScript MX

## I. Свойства:

- Повечето, но не всички вградени типове обекти имат свойства (т.е. стойности, представящи характеристиките на даден обект). Например, ако имаме един мотор, то той има свойства като модел, марка, мощност, цвят. Ако означим мотора като обект на ActionScript, можем да го дефинираме по следния начин:

```
motor.color = "blue";
motor.mark = "Honda";
motor.konskisili = 220;
```

- Един от обектите, които притежават предвари-телно зададени свойства е MovieClip. Чрез тях можем да настройваме прозрачността му, видимост, хоризонтална и вертикална позиция, да осъществим ротация, транслация...

# Същност, свойства и употреба на обектите в ActionScript MX

I. **Свойства:**

- Различните стойности на свойствата на различни обекти може да използвате за задаване стойности на нещо друго. Например дължината на низа е свойство на обекта String. Това ни помага по много лек начин да получим тази стойност за дадена поредица от символи:

```
niz = "MyString123";
lengthOfName = name.length;
// на 1-вия ред създаваме променлива niz
// на която присвояваме низ. А на втория
// дължината на горния низ става стойност
// за променливата lengthOfName, т.е. 11
```

- Според конвенцията за ActionScript MX, имената на свойствата на филми трябва да започват със символа: "_" (_alpha, _rotation…), но поради нуждата от съвместимост с по-старите версии на езика☺ това правило не винаги се спазва☹...

# Същност, свойства и употреба на обектите в ActionScript MX

I. **Свойства:**

- Като свойства на обектите могат да се използват масиви, променливи, както и други обекти.

II. **Методи:**

- В ActionScript методите представляват функциите, които може да изпълнява даден обект. Ако вземем предвид един филм, то методите са: запис (record), възпроизвеждане (play), пауза (pause), спиране (stop), превъртане (rewind), и др. Като записваме първо името на метода, последвано от кръгли скоби “()”...
- За да извикаме конкретен метод на обект, трябва да напишем първо името на обект (или негова инстанция), след това “.” и после името на метода.

```
myMovie.play();
myMovie.pause();
```

# Същност, свойства и употреба на обектите в ActionScript MX

**II. Методи:**

- Като за почти всеки метод, в скобите можем да включим определен брой параметри, с които да манипулираме филмчето по уникален начин.
- Например към филмчето `myMovie` можем да прикрепим следния скрипт:

```
myMovie.record (9, 8.30 pm, 12.45 pm,
January 31)
```

// тук за разделител между отделните параметри се
// използва “,”-та.

- В ActionScript отделните параметри можем да зададем по два основни начина: твърдо (както в примера по-горе) и чрез динамични стойности (с променливи). Също за параметър можем да използваме и други методи.

# Същност, свойства и употреба на обектите в ActionScript MX

## II. Методи:

- Всеки обект в ActionScript има уникален брой методи-отговарящи на специфичните функции, които той изпълнява.
  За MovieClip част от тях са:
  `play(), nextFrame(), startDrag().`
  За обекта String:
  `indexOf(), substr(), toLowerCase().`
  За Color: `setRGB(), getRGB(), setTransform().`
  За обекта Sound:
  `setVolume(), getPan(), start().`
- С методите на обектите тук можем да изпълняваме различни видове задачи.

# Същност, свойства и употреба на обектите в ActionScript MX

## II. Методи:

- Например:
  - взимане (извличане) на стойност
  - задаване на стойност
  - конвертиране на резултат или вход от един тип в друг
  - стартиране или деактивиране на даден обект
  - обработка на текст, друг обект
- *Забележка:* По принцип не е задължително даден обект да съдържа някакви свойства или функции, но без тях става неизползваем и по-скоро даже ненужен.

# Типове обекти

Обектите се организират в класове, представляващи съвкупност от обекти с общи свойства и методи. Например всички хора на Земята могат да се разгледат като инстанции на класа `Person`. Т.е. въпреки нашата уникалност, ти ние имаме някакви общи характеристики. Свойства като цвят на коса, очи, височина; и методи (ходене, гледане, пипане). Така всяка нова инстанция, която създаваме, е инстанция на обекта MovieClip и наследява всички свойства и методи, достъпни за този клас от обекти.

- Основните класове обекти в ActionScript са:
  - MovieClip
  - String
  - Color
  - Sound

# Типове обекти

- Но в повечето проекти ни се налага да използва множество инстанции и то на разнотипни класове обекти. Например, ако трябва да имаме различни низове, то всеки от тях се счита за инстанция на обекта String.
- От своя страна други обекти са универсални или глобални, което означава, че в приложението, което разработваме може да съдържа само една инстанция от тях.
- Пример за това е обектът Mouse, един от методите, на който е да управлява видимостта на курсора. Тъй като курсорът е само един, не можете да създадете негови инстанции, а използвате методите, достъпни за него, за да манипулирате мишката.

# Типове обекти

- Инстанция на обект се създава по два основни начина. Като първият е достъпен само за основните класове обекти и се изразява в просто издърпване на обекта на сцената (това е валидно например за MovieClip).
- За да създадем инстанции за останалите обекти е необходимо да използваме така наречените конструктори. Те указват на Flash да създаде съответната инстанция. Синтаксисът им е следният:

```
nameOfInstance = nameOfObject();
```

- Например, ако искате да създадете инстанция на обекта Sound с конструктор, той би изглеждал така:

```
nameOfSoundInstance = new Sound();
```

# Типове обекти

- Имената на обекти спазват същите конвенции за имена като променливите, т.е. Те не могат да съдържат интервали, специални знаци или цифри като първи знак.
- В ActionScript MX имаме възможност да присъединяваме суфикси към имена на обекти, за да улесним разпознаването на кода в панела Actions. Така вместо да именуваме дадена инстанция на филмов клип `myMovieClip`, можем да я запишем по следния начин: `myMovieClip_mc`. Така тази конвенция ни позволява панелът Actions да определи кога обектът от определен тип има скрипт и по този начин бързо да ни осигури падащия списък от съответните свойства и методи.

# Типове обекти

- В `Actions Inspector->` панела `Actions` под книгата Objects, можете да осъществите достъп до обектите на Flash, всеки от които се съдържа в една от
следните подкатегории:
  - Core (Основни)
  - Movie (Филм)
  - Client/Server (Клиент/Сървър)
  - Authoring (Авторски)
- В следващите слайдове ще разгледаме различните класове обекти, достъпни в ActionScript и кои от тях са глобални или трябва да създаваме индивидуална инстанция.

# Типове обекти

- Обект Accessibilyty (глобален)
  В този обект се съдържа информация (само за четене) относно възможността на компютъра на потребителя да използва четец на екран (screen reader). При положителен резултат връща истина, иначе лъжа.
- Обект Array (инстанции)
  Масивът е място за съхранение на множество от еднотипни данни. Към тях може да се обръщаме с помощта на поредния номер на съответната данна.
  Пример:
```
cake = new Array();
cakeType[0] = "Karamelen";
cakeType[1] = "Shokoladov";
cakeType[2] = "S kaysiev krem";
```
  В този обект имаме полезни методи като: добавяне на нов елемент, премахване, сортиране.

# Типове обекти

- Обект Boolean (инстанции)
  Той съхранява две стойности: истина или лъжа (`true` или `false`) . Инстанция създаваме с помощта на конструктора `Boolean` или с оператора за присвояване "=" .
  Пример:
  ```
  toggle = new Boolean ();
  ```
  или
  ```
  toggle = false;
  ```
  // действието на двете е идентично
- Обект Button (инстанции)
  При поставяне на бутон на сцената, ние създаваме инстанция на обекта Button. (Само още два обекта създават своите инстанции така- чрез drop-down, това са MovieClip и TextField).

# Типове обекти

- Обект Capabilities (глобален)
  Тук се съдържат характеристики за компютъра на потребителя на нашите програмки-филмчета. Като разделителна способност на екрана или способността да възпроизвежда звук PC-то.
  Пример:
```
myVariable =
System.capabilities.screenResolutionY;
// тук съхраняваме вертикалната разделителна
// способност на потребителския компютър
```
- Обект Color (инстанции)
  Този обект може да използваме за динамична промяна на цвета на филмов клип. Когато създаваме обект, трябва да го определим към конкретен клип. След това с помощта на методите на Color може да променяме цветовете. Такъв обект създаваме като използваме конструктор:
```
myColor = new Color (pathToTimeLine);
```

# Типове обекти

- Обект Date (инстанции)
  С този обект имаме възможност да вземем текущата дата, месец, година, час като локално или глобално време, по Гринуич и т.н.
  Пример:
```
now = new Date ();
LargeNumber = now.getTime();
```
  // променливата присвоява определен брой милисекунди
  // след 00.00 h на 1-ви Януари 1970 г. от текущата дата.
- Обект Key (глобален)
  Определя състоянието на клавишите от клавиатурата.
- Обект LoadVars (инстанции)
  С този обект и с помощта на неговите методи може да зареждаме данни от външни източници.
  Пример:
```
myObj = new LoadVars ();
myObj.load("http://www.fmi-plovdiv.org");
```

# Типове обекти

- Обект Math (глобален)
  С негова помощ осъществаваме различни математически изчисления.
  Пример: `positiveNumber = Math.abs(-6);`
  // конвертира абсолютната стойност на обекта Math в
  // положително число.
- Обект Mouse (глобален)
  Управлява видимостта на курсора, позволява ни да задваме слушатели за проследяване активността на мишката.
  Пример: `Mouse.hide();`
  // това скрива курсора на мишката от потребителя, но сама-
  // та мишка си е активна...
- Обект MovieClip (инстанции)
  Това е един от най-популярните обекти. Можем да създаваме негови инстанции по два основни начина: (1) като ги поставяме на сцената или (2) чрез методите `createEmptyMovieClip()` и `duplicateMovieClip()`

# Типове обекти

- Обект Object (инстанции)
  Това е коренният обект в ActionScript. Синтаксисът за нрговото създаване е:

```
imena = new Object();
imena.kotka = "Lyki";
```

- Обект Selection (глобален)
  Този обект се използва основно за извличане на информация или задаване характеристики за селектирани елементи във филмите. По-често се използва за текст в текстови полета. Когато курсора се намира в дадено текстово поле се казва, че е на фокус. Това е полезно, когато искаме примерно да укажем кое поле е на фокус в момента или за да селектираме конкретни парчета текст в поле, така, че то да се управлява по някакъв начин.
  Пример: `Selection.setFocus("Sobstveno ime");`

# Типове обекти

- Обект Sound (инстанции)
  Той се използва за управление на звука, повече за него ще кажем в следващите лекции.
- Обект String (глобален)
  Използваме го за обработка на характеристиките на сцената.
  Пример: `Stage.height;`
  // връща височината на сцената в пиксели...
- Обект System (глобален)
  С него можете да получите данни за компютъра на потребителя като операционна система, използван естествен език плюс всички свойства на обекта Capabilities. Едно от неговите свойства е низ, наречен `ServerString`, който съдържа списък от възможностите на системата. Този списък може в последствие да се изпрати на сървър и да се съхрани или използва получената информация. За достъп до този низ може да използвате: `System.capabilities.String`

# Типове обекти

- Обект TextField (инстанции)
  Обектите от този клас използваме за форматиране на текста. След като сте създали текстово поле, може да използвате метода `setTextFormat()` или метода `setNewFormat()` на обекта TextField.
- Обект XML (инстанции)
- Обект XMLSocket (инстанции)
  С него можете да получите данни за компютъра на потребителя като операционна система, използван естествен език плюс всички свойства на обекта Capabilities. Едно от неговите свойства е низ, наречен `ServerString`, който съдържа списък от възможностите на системата. Този списък може в последствие да се изпрати на сървър и да се съхрани или използва получената информация. За достъп до този низ може да използвате: `System.capabilities.String`

# Използване на обекта Color

В предните слайдове поясних, че за да използвате този обект, трябва да го създадете със съответния конструктор: `myColor = Color (bluza);` // така създадохме инстанция `myColor` на `Color` и я свързахме с инстанция на филмов клип с име `bluza`.

- Тази инстанция можем да създадем навсякъде, а за параметър на конструктора обикновено се задава пътя до филмчето, на което ще въздействаме. В случай, че искате да създадете инстанция от вътрешността на филмов клип, който ще се обръща към себе си трябва да използвате ключовата дума `this`.
  Пример: `myColor = new Color (this);`
  // това създава нова инстанция на обекта Color в инстанцията
  // на филмов клип, който съдържа този скрипт
- Най-често използвания метод на този обект е: `setRGB()`.

# Използване на обекта Color

- Пример: `myColor = new Color (bluza);`
  `myColor.setRGB (0x000066);`
  Този скрипт след като създаде инстанцията на обекта `Color` променя цвета на инстанцията на филмчето `bluza` на тъмносин. Методът `setRGB()` приема за параметър (0x) следван от шестнадесетична стойност на цвят. Параметърът 0x е запазена знакова комбинация, която указва на Flash, че това което го следва е 16-но число.
- Разбира се не е задължително да знаете подробно цветовете като шестнадесетични стойности. Имате възможност динамично да конвертирате цвят в RGB към шестнадесетичното му съответствие, ако разбира се знаете RGB комбинацията;(). Това става с помощта на метод на обекта `Number: parseInt()`.
- Малко примери...

# Използване на обекта Key за добавяне на интерактивност

- Обектът Key се използва за прихващане на събития на натискане на клавиш от потребителя (т.е. Взаимодействие на потребителя със клавиатурата). И по-точно:
  - да определите дали даден клавиш е натиснат в момента
  - да определите кой е бил последния натиснат клавиш
  - да вземете стойността на кода на последния натиснат клавиш
  - добавите слушател за събитие...
  - определите дали ч включен или не определен клавиш (например Caps Lock)

# Използване на обекта Key за добавяне на интерактивност

- Тъй като този обект е глобален, то за него не можете да създадете инстанция. Най-честата му употреба е за проверка състояние на клавиш. За да направите това се използва например следния код:

```
Key.isDown (Key.TAB);
// тук правим проверка за клавиша TAB, като можем да зададем и неговия цифров еквивалент (ASCII кода му):
    Key.isDown (9);
```

- Отново упражнение...

# Работа с обектите String и Selection

- Един от най-често използваните обекти е String. Той използва методи, които са полезни за модификацията и създаването на низове.

```
message = "Tova e niz v ActionScript";
```

Всъщност така горе създадохме променлива message като инстанция на обекта String, т.е. Вече може да използваме неговите методи за обработка на нейната стойност. Можем да създадем инстанция още с конструктор и чрез създаване на текстово поле (`имеНаТекстовоПоле.text`).

- По-важни методи на обекта String:

- `toUpperCase()` - всички букви стават главни...
- `toLowerCase()` - всички букви стават малки...
- `indexOf()` - открива първо срещане на знак или група от знаци в низ. Връща число, съответстващо на номера на индекса на буквата в низа (започва да брои от 0)

# Работа с обектите String и Selection

- Пример:

```
message.text = "Tova e niz v ActionScript";
pyrvoSr = message.text.indexOf("e");
// Това ще върне за резултат: 5.
```

- Ако искате да проверите дължина на низ, може да използвате свойството `length`

Пример:

```
postCode.text = "4000";
postLength = postCode.text.length;
if (postLength == 4) {
// Правилна дължина на кода
} else {
//неправилен пощенски код
}
```

# Работа с обектите String и Selection

- От своя страна обектът Selection ни позволява да управлява-ме текущо фокусираното текстово поле, включително оцветяване на текст, задаване текуща позиция на текстов курсор и ред други. Едновременно на фокус може да бъде само едно текстово поле, никога не може да има повече от един обект Selection.
- Фокусирането на поле от потребител става като той щракне върху полето, но може и да използваме метода:`setFocus()`
- Един от последните методи на този обект ни дава възможност за динамично осветяване на части от текста – без помощта на потребителя, включващ два параметъра (индекса- знака, от който започва селекцията, и индекса, където завършва):

```
Selection.setSelection (7, 10);
```

- В следващото упражнение ще направим прост текстов редактор, за да приложим казаното до тук...

# Работа с функции

# Създаване на функции и добавяне на параметри към тях

- В проектите ни често се налага да пишем едни и същи парчета код повече от веднъж. За да направим използването им по-лесно, можем да използваме функции, а действието,с помощта на което изпълняваме функцията се нарича извикване на функция.
- В ActionScript преди да използваме функция, логично трябва да я създадем или дефинираме. Това може да го направим като използваме два възможни синтаксиса:

  Синтаксис 1:

  *той е и най-често използвания начин за създаване на функция, съответно именно него ще използвам по-нататък в лекцията.*

```
function myFunkcia (параметър1, параметър2 и т.н){
     // действия;
}
```

Декларацията на всяка функция започва с ключовата дума:

`function`

# Създаване на функции и добавяне на параметри към тях

- След ключовата дума function веднага следва името на функцията, то може да е всичко, което преценим, стига да следва конвенциите за имената.
- След името на функцията в кръглите скоби можем да сложим желани параметри или да оставим тази област празна. Ако сте избрали последния вариант, ще създадете функция, която се изпълнява по един и същи начин при всяко извикване. Съдържа ли параметри, изпълнението и ще е уникално за всяка различна стойност на параметъра. Може да предавате произволен брой параметри по принцип.
- Във фигурните скоби слагате тялото на функцията, тоест там се поставят действията, които искаме да изпълнява нашата функция.
- Можем да създадем скелет на функция (т.е. Такава, която не съдържа още действия) като изберем: `Actions -> User-Defined Functions` в панела Actions.

# Създаване на функции и добавяне на параметри към тях

- Синтаксис 1:
  *Вторият начин за създаване на функции е:*

```
myFunkcia = function (параметър1, параметър2 и т.н)
     { /* действия;*/}
```

- Този втори синтаксис се използва най-често, за да създадем функция динамично или за да дефинираме наш собствен метод на потребителски клас. Тук разликата с първия синтаксис е, че първо записваме името на функцията, а стойността се присвоява с оператора за тази цел: “=”.
- Можем да зададем по такъв начин функция само като го въведем ръчно в експерт режим на Actions панела.
- Ако нашата функция не съдържа параметри можем да я зададем с помощта на следния синтаксис: `myFuncia();`

# Създаване на функции и добавяне на параметри към тях

- Но в случай, че сме я дефинирали да приема параметрична информация, то синтаксиса за извикване е следния:

```
myFuncia(параметър1, параметър2);
```

- В горните примери се приема, че функцията и нейното извикване се намират в една и съща времедиаграма. Но в случай, че това не е така и искаме да извикаме функция на конкретна времедиаграма, трябва да поставим целевия път до тази времедиаграма пред извикването на функцията по следния начин:

```
_root.MyClip1.MyClip2. myFuncia();
```

- Може да извикаме една функция и динамично, на базата някаква стойност (например: `_root[aVariableName];`). Тоест, в случай че `aVariableName` има стойност “saySomething”, извикването и реално ще изглежда така: _root. saySomething();

# Създаване на функции и добавяне на параметри към тях

- Примерен синтаксис на функция, която приема параметри:

```
function AverageStoiinost (ObstaStoinost,
     ObshtBroii) {
     srednoaritmetichno =
     ObstaStoinost/ObshtBroii;
 }
```

// Ако искаме да я извикаме: `AverageStoiinost(24,4)`

- Когато извикваме функция в ActionScript се създава временен масив, наричан от системата `arguments`. Той съдържа всички предадени във функцията параметри- дори ако не сме задали такива при дефинирането на функцията. Ако искаме да осъществим достъп до този масив, може да го направим по следния начин:

```
function traceNames (){
trace("Tazi funkciq e priela" + arguments.length +
"argumenta");
trace("Stoiinostta na 1-viq argument e:" + arguments[0]);
trace("Stoiinostta na 2-riq argument e:"+ arguments[1]);}
traceNames("Vanya", "Mitko");
```

# Създаване на функции и добавяне на параметри към тях

- Като резултат от функцията, в прозореца output ще получим следното:

```
Tazi funkciq e priela 2 argumenta.
Stoiinostta na 1-viq argument e: Vanya
Stoiinostta na 2-riq argument e: Mitko
```

- Осъществаването на достъп до горния масив ни дава възможност да създаваме функции, които могат да адаптират функционалността си според броя на предаваните в тях параметри.

# Използване на локалнипроменливи и създава-не на функции, връщащи резултат.

- Достъп до променливите, които създавахме до сега, можехме да осъществим от всяко място и по всяко време, във всеки скрипт от нашия филм. Но използваме ли локални променливи, то те имат обхват само в границата на функцията, за която са дефинирани. И се унищожават при приключване на действието на функцията.
- Локалните променливи не са абсолютно задължителни при създаването на скриптове, но стават добра практика в случай, че имаме по-голямо приложение и броя на променливите нарасне. С тях можем да спестим памет.
- Локалните променливи в дефиницията на една функция могат да имат едни и същи имена с други от друга функция дори и да са в една и съща времедиаграма.

# Използване на локалнипроменливи и създава-не на функции, връщащи резултат.

- Тук декларацията на локални променливи става ръчно с ключовата дума `var`:

```
var myVar = "promenliva";
```

- Можем едновременно да декларираме множество променливи по следния начин:

```
var ime = "Maria", familia = "Dimitrova",
site = "www.maria.net";
```

- Можем да създаваме и функции, връщащи резултати с `return.`:

```
return ime;
```

- Можем да използваме действието `return` , за да връщаме произволни типове данни, например стойности на променливи, масиви или други обекти.

- А сега отново малко практика...